Autonomous UAV Based Search Operations Using Constrained Sampling Evolutionary Algorithms

Gervasio Varela, Pilar Caamaño, Felix Orjales,
Álvaro Deibe, Fernando López-Peña, Richard J. Duro
Integrated Group for Engineering Research
University of A Coruña
Ferrol, Spain

gvarela@udc.es, pcsobrino@udc.es, felix.orjales@udc.es,
adeibe@udc.es, flop@udc.es, richard@udc.es

Abstract

This paper introduces and studies the application of Constrained Sampling Evolutionary Algorithms in the framework of an UAV based search and rescue scenario. These algorithms have been developed as a way to harness the power of Evolutionary Algorithms (EA) when operating in complex, noisy, multimodal optimization problems and transfer the advantages of their approach to real time real world problems that can be transformed into search and optimization challenges. These types of problems are denoted as Constrained Sampling problems and are characterized by the fact that the physical limitations of reality do not allow for an instantaneous determination of the fitness of the points present in the population that must be evolved. A general approach to address these problems is presented and a particular implementation using Differential Evolution as an example of CS-EA is created and evaluated using teams of UAVs in search and rescue missions. The results are compared to those of a Swarm Intelligence based strategy in the same type of problem as this approach has been widely used within the UAV path planning field in different variants by many authors.

1.Introduction

In the last few years the use of Unmanned Aerial Vehicles (UAVs) for an ever increasing variety of applications has grown remarkably. These small aircraft can be controlled remotely or programmed to fly in an autonomous way. They have been used for different types of applications as individual entities, both in military and civil tasks [1][2][3]. Research related to this type of systems is being undertaken in many areas going from the design of more efficient aircraft aimed at specific applications [4], to the development of improved control electronics that provide for better autonomous behaviours, to optimized path planning strategies [5][6], or to opening of new application domains [7]. In many application scenarios, the use of several autonomous UAVs working together can speed-up the sensing or exploration process by splitting the task into different parts that can be carried out simultaneously. In addition, the use of teams can increase accuracy by fusing information that has been obtained from different sources. In some other cases, collaboration enables teams of UAVs, when working together, to complete a task that would be impossible with only one UAV. Thus collaborative UAV systems are starting to appear in the scene as a very interesting alternative to other more traditional methods. This field is still in its infancy and many exciting new approaches are being explored for different applications [8]. Examples of these are works related to trajectory planning in UAV teams [9][10], real time target tracking [3], and many others [11][12].

In this paper the applicability Evolutionary Algorithms (EAs) to the real time coordination of teams of UAVs while performing search operations is investigated. They will be compared to Particle Swarm [13] based Algorithms which have been extensively used in the field due to their completely distributed approach where every airplane operates autonomously and collaborates with surrounding airplanes to explore the environment and accomplish the search mission. Swarm Intelligence based algorithms have been extensively used for this purpose, see for instance [14][15][16] and [17] for some examples, including the work presented by the authors in [18].

The reason for choosing evolutionary algorithms over other search methods as the second strategy for this study is twofold. On one hand, these algorithms are intrinsically multipoint, which in real systems can be translated as multiunit. On the other, the exploration vs. exploitation balance that is necessary to produce efficient search procedures is easy to regulate in some of these algorithms, helping to improve the resulting operation of the system. Evolutionary Algorithms have been developed and used extensively to solve many types of optimization problems, especially when the fitness landscapes are complicated and hard for more traditional and non-stochastic techniques. However, unlike Swarm Intelligence based strategies, they have seldom been used for real time real world applications due to their computational cost. In addition, when they have been used to this end, the algorithms have been made to work as an optimizer for certain parameters of the real time system and not really as an intrinsic part of its operation. This is the case of some of the UAV related evolutionary algorithm applications found in the literature which are always focused on solving particular aspects, such as design optimization [19], or path planning [20] .

Nevertheless, we believe there is a very interesting niche in which evolutionary algorithms can perform well being a part of the operation of the system. This is in problems that benefit from the use of large numbers of units working together and, in particular, that of the application of evolutionary algorithms within coordination and cooperation strategies in systems made up of multiple units. Examples of these types of problems are related with the use of teams of UAVs in search operations, mapping strategies, environmental monitoring, etc. Many of these problems are intrinsically or can easily be converted into exploration or search problems, that is, a set of units (team) have to work together to find one or several objectives. Obviously, the efficiency of a search procedure depends on how well the different units use the clues the team gathers from the environment in order to better choose the next areas or points that must be explored. In other words, the key of a good search strategy is to be able to correctly infer where the target may be from as few samples of the environment as possible.

This is very similar to exploring unknown search or solution spaces in optimization problems in order to find their optimum or optima. In fact, it is the exact same problem except for the fact that this exploration is being carried out by a set of real entities or units that due to the fact that they are real, are constrained by the limitations of reality (they cannot be omnipresent, or change positions instantaneously, etc.). Therefore one can hypothesize that the same type of algorithms can be used to coordinate the search the teams perform as those used when searching in optimization problems. As an extension, one could assume that the same exploration/exploitation types of criteria could be applicable to these types of problems if the operators of the algorithms can be adapted so that the constraints of reality are met in their operation. Here we are going to denote the type of problems where the sampling of the search space is limited by the physics of reality as Constrained Sampling (CS) problems. The coordination of UAVs in search operations is clearly within the scope of CS problems.

A lot of work has been carried out in the field of function optimization to address difficult solution spaces. In fact, many algorithms have been proposed that provide good optimization solutions in quite complex search spaces that are hard either due to their ruggedness, which makes it very difficult for gradient based or other types of single point deterministic algorithms to find the optima, or to their flatness, which provides very little information or clues for the algorithms to choose where to explore next, leading to a basically random search strategy. In particular, certain types of modern evolutionary algorithms have been shown to be very adaptable to operate in such conditions; examples are CMA-ES [20], DE [21], and others. However, these types of algorithms cannot be directly applied as an intrinsic part of the operation to the coordination of the search strategies of real multiunit teams, that is, they are not directly applicable to Real World-Real Time problems as their formulation does not comply with the constraints imposed by using real units in the real world, that is, to the CS nature of these problems.

Usually, most evolutionary algorithms take samples from different points in the search space and using the information provided by these samples decide on the next points that must be sampled. These new samples usually turn out to be quite far from the original ones implying that a real unit would have to move to this position and measure the sample. This takes time and is not very efficient. Consequently, in this paper, and in terms of evolution, we are going to adapt a well-known evolutionary based optimization strategy, the Differential Evolution algorithm, to the CS nature of the problem with the objective of producing an efficient and adaptive search procedure for teams of

real world UAVs and of studying the capabilities of this type of approaches. The results of this approach will be compared to those provided by a swarm intelligence based strategy in order to determine its virtues and drawbacks in terms of one of the most commonly used approaches.

Summarizing, we study the application of two strategies to a certain type of CS problems. On one hand we have implemented and tested a swarm intelligence based search algorithm that operates in a completely distributed fashion. On the other, we propose a general strategy for the adaptation of Evolutionary Algorithms to their operation on real systems working in real time. It is to their operation in Constrained Sampling problems. In particular, in the experiments carried out we will present a new version of the traditional Differential Evolution (DE) Algorithm. This version of the algorithm has been called the Constrained Sampling Differential Evolution Algorithm (CS-DE).

The study will be carried out considering a real example related to the use of UAV teams for finding several survivors of shipwrecks in a very large area. That is, the airplanes fly autonomously trying to reach the spatial points that the Differential Evolution or the Swarm Intelligence based algorithms, using the sensor information provided by the airplanes in the previously visited points, decide that must be explored. Both approaches are intended for implementation in real search operations, so they must be realizable, fast enough to cope with the requirements arising from running in real time, and robust enough to handle the uncertainties due to changes in environmental parameters or in the target itself that commonly occur in any real world application. In fact, to ensure realism, all of the simulations and executions for the study and evaluation of the algorithms are performed using tools that have been calibrated to the parameters and idiosyncrasies of the real UAV team that will be employed in real tests.

The remainder of the paper is structured as follows. The implementation and details on the two different collaborative search strategies that have been developed are discussed first in Section 2. Then Section 3 presents a 3D simulator able to realistically represent the behavior of the UAVs and which has been developed and employed to analyze the performances of different coordination strategies. The results of several experiments using the system are presented and discussed in Section 4. Finally, Section 5 provides some conclusions and a description of future lines of work that are open.

2. Cooperation strategies for finding targets

This section presents the two different exploration strategies considered in this work and their adaptation to Constrained Sampling Problems. As previously indicated, the first one is a swarm intelligence based search algorithm that operates in a completely distributed fashion where each agent follows a simple set of rules. By interacting with the environment and the surrounding neighbors an emergent behavior arises that allows the team to find the objective. The second one is a completely different approach that attempts to benefit from the capabilities of evolutionary algorithms to make use of the information obtained from previous samples in order to appropriately direct the search towards the targets. In this case, the evolutionary algorithms (EA), are fed with the data coming from the environment. The information provided by the EA is used to guide the search towards promising areas. These strategies can be made more or less centralized depending on the particular EA that is used and on whether the particular operators of the EA can be run in a distributed fashion, that is, without operations that require information from the whole population. However, as commented in the introduction, any EA that we want to use on real world real time problems must be adapted to the constraints of reality, which does not allow sampling any point at any time. We have called these EA variations "Constrained Sampling Evolutionary Algorithms" or CS-EAs for short. The basic set up of the CS-EA developed in this work allows coupling any type of Evolutionary Algorithm and in the particular example considered here involving UAVs, the plane controllers, in addition to their client implementation of a particular CS-EA, implement local collision avoidance, minimum height of flight and stay within a given area behaviors that override and constrain their operation to preserve their integrity and safety.

2.1. Swarm intelligence based approach

The swarm intelligence based search strategy uses a completely distributed approach where every physical agent operates autonomously and collaborates with surrounding neighbors to explore the

environment and compile sensing data. In this strategy the agents operate following two phases. A flow diagram of their behavior is displayed in Figure 1.

During the first stage the agents perform an exploration behavior. While they are doing this, they are also sensing the environment seeking data values above a fixed alarm threshold. Also during this stage, the agents start broadcasting the data sensed using their sensors by means of the different communications channels available to them so that other agents of the system can receive these data (at least those that are close enough). If during this state, the agent detects a value that is higher than the alarm threshold value, the agent changes it state to a search state. In any other case, the agent continues with the exploration behavior until it receives or senses a data value above the alarm threshold.

As soon as one of the agents detects a sensing data value above the threshold it enters the search mode. In this mode the agents start collaborating with their neighbors in order to find the data source. As every agent is receiving the data sensed and broadcast by others surrounding it, it uses the data coming from the N nearest neighbors, and its own sensing data, to select a promising direction for continuing its search. Sometimes, in real problems, where the data is noisy, the agents must average a series of sensor data values in order to consolidate what will be the data item it sends to its neighbors. The agent uses the information provided by those around it to determine the surrounding agent with the maximum sensing value. If it is larger than its own sensing value, the agent changes its search direction or path towards the current position of the agent that provides the maximum value in its surroundings. Otherwise the agent continues the search following the current path. This decision about which path to follow is made every T seconds (T is a parameter of the agents that we call 'decision gap'). Thus, an agent will follow a given search path for at least the length of the decision gap, and then it will decide whether this path continues to be promising or not.

When an agent starts detecting data values above the threshold and starts collaborating with surrounding agents, the agent has become a member of a virtual team that is exploring a particular promising area of the environment. Therefore, the expectation is that this strategy should lead to the autonomous emergence of different teams of cooperating agents. A team is created when an agent starts following other surrounding agents because they are reporting positions that are more promising in terms of finding maximum values than its current one. Obviously, any agent can leave a team at any moment, as long as it finds itself a more promising value, thus creating a new possible group, or it detects a different surrounding agent that has a more promising sensing value.

The group behavior is controlled, to some extent, by the decision gap parameter, which can be said to control the exploration vs. exploitation ratio of the search strategy. If this parameter is low, the displacements of the agents between heading changes will be small, making their behavior much more dependent on other agents in their team. As the decision gap is increased, the displacements of the agents between path changes will be larger, thus increasing the exploration capacity of the strategy and, therefore, increasing the probability of finding new promising areas, at the cost of a possible loss of exploitation capabilities, that is, losing precision in the determination of the target position if it is within one of the promising areas it was already exploring.

2.2. Evolutionary Algorithm Based Approach

Unlike the swarm team strategy presented above, the evolutionary one uses a hybrid approach where distributed and centralized control is used. In fact, in general, the degree to which the strategy is distributed depends on the particular operators of the specific Evolutionary Algorithm used in the implementation. Some algorithms, like Differential Evolution [22], for instance, can be made completely distributed if so desired, as its crossover and mutation operators only depend on a small group of individuals and these could be chosen from the close surroundings of an individual. However, other algorithms, such as CMA-ES [21] depend on the calculation of a distribution function that takes into account the fitness values of all of the individuals in the population. Therefore it is not easy, or even convenient in most cases, to completely distribute this algorithm.

Anyway, here we are going to consider the general case of any EA and thus, the approach will be described in terms of a hybrid approach. In this strategy the agents operate in a distributed

manner, they are able to move autonomously throughout the search space avoiding obstacles. As it is an evolutionary based approach there is a need to define a fitness value for each point that is explored. In problems that involve real agents operating in real environments performing search operations, this fitness value will be given by a value provided by a sensor or set of sensors of the agent. That is, if the agents are trying to determine sources of pollution, for instance, the fitness value of a given point is determined by the value sensed by the pollutant sensors, if the agents need to determine the position of an electromagnetic signal, as will be later shown in the examples, the fitness value is given by the value provided by the signal sensors in the agents.

In this strategy, the agent behavior also presents two phases of operation. There is an initial exploration stage that is necessary to spread the agents out throughout the environment before starting with the evolutionary stage. Otherwise, as in most real applications the real physical agents must be deployed from a single or very few points, they search would be biased towards the areas surrounding those points. The exploration stage behavior is different from the case of the swarm strategy. During this stage the agents send the information sensed from the environment to the central controller (or to the controllers of the nearby agents in a more distributed case) but they do not receive any information from it and continue with this exploration or spread out behavior until they spread out over the whole environment. When the central controller (or the distributed controllers) begins receiving sensing data, it instantiates an evolutionary algorithm and it starts feeding the algorithm with the sensed data coming from the agents. The inputs of the algorithm are the positions of the agents and the data value sensed by the relevant sensor or sensors at those positions. The evolutionary algorithm stores the input data in a queue until it has enough data to execute an evolutionary step. This way, the number of searching agents is decoupled from the population necessary to run the evolutionary algorithm. Once the algorithm has enough data (that is, a large enough population of sensed values), it generates a new population of positions that, according to the evolutionary process, are more promising than the current ones. These new points are the result of applying the appropriate evolutionary operators to the parent population. Each new population generated is temporarily stored by the controller, and is used to select promising paths for the agents.

Once the exploration stage finishes, when the agents have spread out, each agent starts taking into account the results from the evolutionary process. Here the goal direction or path of each agent is managed by a central (or distributed) controller. Figure 2 and Figure 3 show the state diagrams of these processes. During the evolution stage, every Z seconds (this parameter can be chosen by the user) each agent sends a message to the central controller requesting a new goal position. When the controller receives a request, it uses the currently stored population resulting from the previous evolutionary step of the evolutionary algorithm to select an available position for the requesting agent. It then removes it from the available positions, to prevent two agents receiving the same goal location, increasing this way the exploration capabilities of the system, and sends the new goal position to the agent. When an agent receives a response to its path request, it seeks to change its current heading in order to try to reach the goal it has received. This path is followed until the *Z* seconds have elapsed and the process starts over again.

As indicated above, there is a stage within the evolutionary approach that requires an evolutionary algorithm to run. In that stage, using a set of positions as population and the corresponding values sensed by the agents at these positions as fitness value, the central controller obtains a new population of promising positions that should be explored. However, most evolutionary algorithms, when they generate new populations they do not take into account the fact that a physical entity must preserve continuity in its trajectory and has limitations and constraints on the kind of movements it can achieve. This implies that they cannot instantaneously (defined as within a time interval t) move to positions that are not contiguous in terms of the time interval or that are not available within its motion constraints. That is, new populations of positions may imply discrete jumps in space or may require movements that are not feasible for the agent. On the other hand, if the algorithms are completely constrained to producing only contiguous positions that respect the motion constraints of the agent, much of their search power is lost. To prevent this problem when applying evolution to real entities, the populations of these evolutionary algorithms must be decoupled from the real entities they are controlling. This is the case in point, where the positions of physical agents that search for a target are evolved. It is often the case that the fitness of these new positions proposed by the algorithm cannot be evaluated because the agents are the only elements of the environment that are able to evaluate a point. Consequently, several modifications

to a traditional EA are needed to solve problems in which the individuals of the EA population are motion constrained.

Based on the common execution cycle of an EA, we have developed a variant called Constrained Sampling – EA (CS-EA). The term "constrained sampling" has to do with the impossibility of moving the individuals that are sampling the search space directly to the positions generated by the EA and, consequently, the impossibility of evaluating the fitness of these positions or individuals every generation. To deal with this problem several modifications are made to an original EA. First of all, the CS-EA algorithm requires two populations for this purpose:

- A population of individuals that represents the position and sensed values in time of the agents that evaluate the search space during their motion. This population will be denoted as the evaluated population. It includes a set of point positions and a value for the fitness (sensed value) of these positions.

- The population generated by the algorithm using the reproduction operators and the fitness information provided by the evaluated population. It corresponds to promising positions of the search space. It will be denoted as the target population.

Each instant of time, the agents send their positions and the values provided by their sensors to the CS-EA, these values will be used as the evaluated individuals of the EA. The CS-EA waits until the size of the evaluated population reaches the CS-EA population size. At this point, the CS-EA executes a generation; during the selection phase the CS-EA uses the evaluated population to select the parent population. With this parent population the CS-EA executes the reproduction operators of the particular EA that is implemented generating a new population of target positions that is stored as the target population. After the reproduction stage, the CS-EA executes the replacement stage. To perform it, the CS-EA has to wait until a new population of evaluated individuals is received from the agents (which usually do not correspond to the population of target individuals produced by the algorithm in the previous stage). The replacement operators compare the parent population used in the reproduction stage with new individuals from the environment to create a new parent population that is used to carry out the processes leading to the next generation of the CS-EA.

As indicated above, due to the motion constraints of the agents, they cannot be moved directly to the positions determined by the target population. Instead, when an agent requests a new goal position from the controller, the CS-EA algorithm sends it this population and the controller decides the new goal position for each agent and the agent starts to try to move towards it. Summarizing, the target population generated using the CS-EA operators guides the motion of the agents along the landscape towards areas that the CS-EA considers promising but this population cannot be used to guide the evolutionary process due to the fact that it is usually impossible to evaluate the fitness value (sensed value) for these positions within the time interval assigned, that is, there is usually no agent close enough to take the measurement. However, as the agents move, they are evaluating positions in the environment in the direction of the target positions and these can be used as evaluated individuals in the CS-EA. The steps of the CS-EA algorithm are shown in the flow diagram of Figure 4.

As previously indicated, the evolutionary approach proposed here could be coupled with any EA. In this work, a Differential Evolution (DE) algorithm will be used as a starting point because of its exploration capabilities. Moreover, due to the features of the target applications that this work is aiming at, in which there will be multiple targets in the search space, out of the large number of variants of this algorithm that can be found in the literature, we have selected one that can handle multimodal problems using the concept of niching.

The basic scheme of the DE algorithm iterates over a population of candidate solutions applying a simple formula that combines information of randomly chosen individuals of the population to generate a new population of potential solutions. Each candidate solution of the population is an n-dimensional vector, being n the number of parameters of the function to optimize. In each generation, during the reproduction stage, the DE algorithm applies the mutation and crossover operators to each individual of the population. The main operator of this algorithm is the mutation operator that it is applied to each individual, also called *target vector,* of the population. The resulting individuals are called mutant individuals. Several mutation strategies have been proposed

for this algorithm, being the most commonly used random mutation (*rand*) and best mutation (*best*). After the mutation stage, the crossover operator is applied. This operator merges the parameters of each *target vector* with the parameters of its corresponding mutated individual. There are also several proposed crossover operators; the most commonly used are *bin* or binomial crossover and *exp* or exponential crossover. The result of crossover is another vector called *trial vector* which is compared to the *target vector* in a greedy manner, i.e., the one with the best fitness value survives as a member of the population of the next generation.

The previous explanation corresponds to the DE base behavior. This basic DE scheme, with the best or random mutation strategy, was proposed for the optimization of functions that present only one global optimum. However there are scenarios, like the one presented in this work, where we require the possibility of locating several optima. To this end a commonly used approach in the field of Evolutionary Algorithms is inspired by the biological concept of niching [23]. Taking advantage of this concept, the EAs preserve the diversity of the population through the formation of niches or sub-populations that explore different and spread-out promising areas of the search space. This allows the algorithm to simultaneously converge to several solutions of the function in the same run. EA niching techniques that can be found in the literature include crowding, fitness sharing, clearing, clustering, parallelization or speciation. In this work we have considered the DE variant for multimodal problems presented in [24] as our base algorithm that will be adapted to CS problems because of its successful performance solving the most commonly used multimodal benchmark functions. It takes advantage of the clustering behavior of the random mutation strategy and incorporates the concept of vicinity by using the information of the closest individual of the population to generate the mutated individual. More specifically, for each individual of the population $x_g^i;\ i = 1, 2 \ldots, NP$, where *NP* is the total number of individuals in the and *g* indicates the generation, the *trial vector* is generated according to the DE/nrand/2/bin scheme. Where *nrand* is the mutation strategy that generates the *mutant vector* (v_{g+1}^i) as follows:

$$v_{g+1}^i = x_g^{NN_i} + F\left(x_g^{r_1} - x_g^{r_2}\right) + F\left(x_g^{r_3} - x_g^{r_4}\right)$$

where $x_g^{NN_i}$ is the closest individual to the current one x_g^i, $r^1, r^2, r^3, r^4 \in \{1, \ldots, NP\}/\{i\}$ are random integers mutually different excluding the current index *i* and *F* is a real parameter called scaling factor. The *trial vector* is obtained by the application of the common binomial crossover operator to the mutant and target vector as follows. For each parameter of the individuals *j*, where *j = 1…n*, the *trial vector* (u_{g+1}^i) is generated as follows:

$$u_j^i = \begin{cases} v_j^i & if\ rand(0,1) < CR\ or\ j == k \\ x_j^i & otherwhise \end{cases}$$

where *CR* is the crossover rate and *k* is a random integer value in [0, *n*) which ensures that at least one parameter of the mutant vector is inherited.

3 Experimental Set-up

As stated above, the approaches presented in this work are intended to be used in real world teams of UAVs. In order to avoid the high costs of performing real test flights, and to be able to extensively test the different approaches under the same conditions and constraints, it was necessary to create a simulation environment where the algorithms could be evaluated. The parameters of the UAV models used within this simulator have been obtained from flight tests of 3 UAVs that have been used later on in the tests presented in the next section. The characteristics of the simulation environment and that of the UAVs used are presented in the following sub-sections

3.1 Simulator

The simulation environment has been implemented as a distributed multi-agent system, in which the planes are autonomous agents that fly in a real or virtual world populated by emitting sources that can be sensed by the planes. One of the main characteristic of this simulation environment is the complete decoupling between the controlling algorithms and the flight dynamics of the planes. This allows us to use different kinds of planes, from simulated planes using Flight Dynamics Model (FDM) software, to real planes remotely controlled by the software. The environment even allows

the simultaneous mixture of different planes (virtual and real), which is a very interesting feature that, in the future, will allow us to perform test flights with a mixed team of real and virtual planes, so that we can test and validate the algorithms with a large number of planes but using a small number of real planes, thus reducing the hassle and cost of the tests.

Figure 5 shows the block diagram of the hardware/software architecture of the simulator. As shown, it is made up of:

- Squadron Controller (SC): It is in charge of providing visualization capabilities and a central control point for those algorithms that require it. It also contains sets of simulated emitting sources, providing simulated sensing values to the virtual planes.

- Virtual Aircraft (VA): It is a virtual representation of a plane in the simulation environment. It has its own behavior that decides the directions that the plane follows. It interacts, through other components, with a simulated or real plane in order to control it in a world with physical constraints, winds, etc.

- Airplane ClosedLoop (AC): It is the connection between a VA and its FDM. It implements the particular logic that interacts with a simulated or real plane, thus allowing the decoupling between the simulation environment and the FDM. In the current implementation of the simulation environment only one AC has been implemented. It uses an instance of the FlightGear flight simulator to simulate the flight dynamics of a plane. The AC is complemented by an Autopilot component that stabilizes and controls the flight of the aircraft, and depends on the FDM used by the aircraft.

- FlightGear Flight Simulator: It is an open-source flight simulator software intended for use in research or academic environments, pilot training and as an industry engineering tool. It supports many different kinds of aircrafts with different FDMs. We have developed an autopilot software tool that controls the throttle, the elevator and the ailerons of the plane to stabilize and move it to follow the direction indicated by the VA behavior.

Summarizing, the simulation environment uses a central control point, the SC, for monitoring and visualization purposes (using the Google Maps API), and for global squadron control in some algorithms. The SC communicates remotely, via UDP/IP, with the VA software, which can be run in remote computers, so that a computation cluster can be used to run multiple planes. This remote operation capability allows running a VA directly in the flight field, connected via radio control to a real plane and remotely connected to the SC via Internet.

For simulating the dynamic behavior of the planes we are using one instance of FlightGear Flight simulator for each simulated plane. Some specific real flight tests were made for each one of them in order to obtain the aerodynamic parameters required by the simulator. Once they are correctly modeled, each instance of FlightGear interacts with a Java program, which implements the Virtual Aircraft agent and runs the behavior of the VA and the Autopilot software. This interaction with the flight simulator is done using the remote control features of FlightGear, which allows remotely accessing and modifying all the FDM properties of a simulated plane. Figure 6 shows a snapshot of the 3D simulator screen with the 2D view using the GoogleMaps API to monitor the positions and routes followed by the planes, and the virtual FPV camera of each plane, which is simulated using one instance of FlightGear per plane.

As the virtual (FlightGear) or real planes move throughout the environment, they receive simulated sensing data from the SC or produce real data. In the case of simulated planes, this data is used by the VA for feeding their virtual sensors. In the case of real planes, they can use this simulated data, or that from their own physical sensors.

Depending on the nature of the algorithm tested, either the control logic will be distributed and will operate in the VAs, being the SC only a mere spectator for providing common sensing data and monitoring capabilities. Or it will be distributed between the VAs and SC, where the SC will act as a controller, sending orders to the planes, which will follow those orders.

3.2 Aerial Vehicles

As initial testing UAV platforms we have considered three different plane platforms to check the practical convenience of several alternative configurations (see Figure 7). The basic platform has wingspan of 1660 mm, a length of1200 mm, a payload of 1.0 kg and up to 40 km range.

This platform is equipped with an autopilot based on a GPS, a miniature Inertial Measurement Unit (IMU), a Pitot speed probe and a barometric altimeter to be able to generate autonomous flight modes, along with a sensor to measure the electromagnetic radio signal strength (RSSI: Radio Signal Strength Indicator) from a set of ground beacons. In addition, it has some space to add different sensors that could be needed to complete different missions. A diagram of the systems in the planes is presented in Figure 8.

For safety reasons in this early stages, these platforms are FPV ("First Person View") capable, in order to allow a human pilot to take control of the plane in potentially dangerous situations. The FPV equipment includes a RC Long Range System (LRS) able to send commands to the UAVs up to 170km in the 869 MHz ISM band and a First Person View (FPV) system with an onboard camera and video downlink in the 2.4 GHz ISM, for visual flight maneuvers.

The data from the onboard sensors, position, speed and other relevant flight information is gathered and sent to the ground station, embedded in the video link, and read in real time from a ground based PC. This information is also superimposed on the video signal with an On Screen Display (OSD), thus the human pilot can fly the plane in instrumental mode if need be.

4 Experimental tests and results

The experimental setup described in the previous section was constructed in order to test the operation of the two coordination strategies of section 2. The task considered in the experiments was that of finding different numbers of shipwrecked people in a large area of the sea. This task is quite costly for conventional means and is very adequate for UAV teams. In this case, as we were just studying the capabilities of the algorithms, it was assumed, as it is commonly the case, that all these shipwrecked people were wearing a life vest provided with a low power radio beacon. Thus the search is based on the detection of the emergency signals given off by these beacons. In a real case the directionality of these radio signals would allow for the detecting devices to point towards the source of emission thus facilitating the search process. This is not the case in many other UAV search or monitoring applications, as for instance in environmental monitoring and pollution source detection, where the measurements are made over a purely scalar quantity –i.e. pollutant concentration- providing no direct directionality information. To be sure that the proposed approaches can work in this sort of applications, the search is carried out without making use of the directionality information.

4.1 Tests flights

As indicated, the purpose of the tests was to study the behavior of the algorithms and this implies a large number of runs of the algorithms under different circumstances. Consequently, all of the tests were run on the simulator and the real flight tests were used to characterize the planes and provide the adequate parameters for the simulators and obtain information on the reality of the radio beacon signals, that is, their strength and reception noise under different conditions so that they could be adequately modeled.

As it is not possible to avoid banking and pitch changes during plane flight, to minimize polarization mismatch losses, as a consequence in this work we decided to use circular polarization antennas, both in the transmitter and receiver. The transmitter antenna is a helical design, and the receiver one is of a Skew Planar Wheel type. Circular polarization has two other advantages over linear polarization, significant in this case. The first one is that circularly polarized signals are less prone to obstacle interferences. They can travel foliage, vegetation and light obstacles with significantly lower losses than linearly polarized ones. This could be significant, for instance, in cases where the objective to be found is lost in a forest. Additionally, circular polarization minimizes multipath phenomena that are common with linear polarization type antennas.

Being the radiation pattern of the antennas used non spherical, contemplating only one of these antennas in the sensor aboard the plane would make the sensor sensitive to the plane orientation. This is an effect to be avoided, because we need to measure the electric field received in the sensor, independent of its orientation. To minimize this issue we opted for the use of three receivers matched with three antennas that are orthogonally oriented to each other. The maximum of the three signal strength indications (RSSI) received is used to represent the received signal power. This sensor arrangement can be seen in the right part of Figure 8.

The data recorded during the test flights is presented in Figure 9 where the x-axis represents time and the two lines represent, respectively, the distance to the beacon and the Relative Received Signal Strength (RSSI). The latter has been modified using a logarithmic correlation in order to achieve the best fit to the former. This correlation has been consequently used in the simulator.

4.2 Results

In order to test the algorithms, a battery of tests has been carried out using the previously described simulation environment and the data extracted and modeled from the flights performed using the real planes. These tests have consisted in the search for different numbers of shipwrecked people within a 100 km^2 area of the sea using the two algorithms and different sizes of UAV squadrons.

In order to simulate the values sensed by the planes, as indicated in the previous section, a function has been modeled to reasonably mimic the behavior of the radiation pattern of the antennas extracted from the real flights performed with our test planes. This function has been used to feed the virtual sensors in the planes and, for this first stage of development, we have decided to model only static emitting sources. That is, it is assumed that the shipwrecked people do not move while the planes are searching. Nonetheless, in order to measure the performance of the algorithms with respect to the distance and position of the emitting sources in the area, a set of random positions have been defined and used in the different tests.

To characterize the behavior of the algorithms according the nature of the sensed values, two different versions of the virtual emitting source function have been used. One without noise, that could, in principle, be easily solved using gradient based algorithms, and one with noise (the noise levels where modeled after the real data extracted from the real planes) that would theoretically be more difficult for gradient based algorithms.

In order to compare the algorithms in terms of time taken to accomplish the task of finding N emitting sources, the planes are equipped with a virtual computer vision system that allows them to visually confirm the presence of a shipwrecked person in their surroundings (we are assuming that they can "see" the targets when they are within a radius of 100 meters from the position of the plane). Thus, the algorithms are run until all the emitting sources present are visually confirmed by a plane. Furthermore, in order to minimize the influence of random successes due to planes that visually confirm the sources during the exploration stage, we have decided to require that at least two planes confirm the position of an emitting source in order to consider it as found.

To set-up the CS-EA algorithm it is necessary to configure the underlying DE used. The configuration parameters are those recommended in [24]. Specifically, we have used a value of 0.5 for the F parameter, 0.9 for the crossover rate (CR) and a population of 100 individuals. Moreover, the strategy used to select a promising objective is based on choosing a random point from the target population of the algorithm that is in the detection range of the planes. This detection range is taken as infinite in distance but limited to an angle of 45 degrees in its heading direction. In the case of the swarm approach, the configuration parameters are the decision gap (T) the value of which is set to one minute and the number of closest neighbors taken in to account in the selection of the heading direction (N), which is set to 3.

For each test, the times required to find each existing emitting source have been recorded in order to be able to compare the algorithms in terms of time and in terms of how many sources have been detected, as not all the combinations of squadron sizes and emitting source numbers are solved by both algorithms. Finally, the search has been limited to a maximum of sixty minutes. More time than that will exceed the flight autonomy of the planes and in real conditions in the Atlantic, the

average survival time is usually no more than 50 minutes due to hypothermia. As an example of the test runs, Figure 10 displays a sequence of snapshots from the simulator running the CS-EA algorithm with five planes and two shipwrecked people. The planes are shown as arrows, with their trajectories drawn as colored lines. The stars mark the target population of geographical points generated by the CS-EA algorithm in that generation. Initially (a) the algorithm generates a highly dispersed population. As the planes start exploring the space, the target population of the algorithm starts concentrating around the areas where the agents have sensed the most promising values (b, c and d). As the simulation progresses, the algorithm starts to exploit the already collected information by concentrating the target population near the emitting beacons of the shipwrecked people (e and f). As the planes tend to move towards the target population, the concentration of the target population obtained by the evolutionary algorithm over the areas where the shipwrecked people are located end up leading the planes to fly over these areas (g and h).

The graphs in Figure 11 show the results provided by the plane teams in the different tasks used to evaluate the performance of the CS-DE as compared to the swarm approach. Obviously, as the number of shipwrecked people increases the task becomes harder in all cases. Take into account that the area being explored corresponds to 100 Km^2 of sea and the number of planes is quite small (15 at most). This can be clearly appreciated in Figure 11 where, in for all three squadron sizes, as the number of objectives increases, the time it takes the CS-EA to find them increases on average. This is more remarkable in the case of the swarm based approach where the number of successful runs decreases drastically when the tasks involve two or three shipwrecked people, especially in the case of the smaller squadron (5 planes). In this case the swarm based approach has no successful runs when there are more than 1 shipwrecked person. This is because it is quite complicated for the algorithm to be able to divide the units into teams with so few units available. However, when the task is completed with a successful result, as in the case of the one-target runs, this approach can be marginally faster than the CS-EA due to its higher exploitative capabilities. Once one of the members of the swarm teams finds a shipwrecked person, the other members start to follow it and reach this shipwrecked person in a relatively short amount of time, often at the cost of leaving areas unexplored as shown in Figure 12 which shows a sequence of snapshots from the simulator running the swarm based control algorithm with five agents and two shipwrecked people. As illustrated by the results, with five agents, the swarm based algorithm is only able to find one shipwrecked person. This sequence illustrates the process, first the agents spread out over the environment while exploring it (a). As the agents sense values above a specified threshold, other agents start following them (b and c), until all of them are near the emitting source (d). Notice that it has missed two of the shipwrecked people.

The evolutionary approach, on the other hand, turns out to be more balanced in its exploitation vs. exploration capabilities. This may be a slight disadvantage when there is just one shipwrecked person, but it is definitely advantageous when the number of shipwrecked individuals increases as the chance of missing one of them decreases drastically. This can be seen in the results of the executions with two and three shipwrecked people. By increasing the number of planes in the team (Figure 11.b) the results of the swarm based approach improve as its explorative capabilities also increase with the number of planes due to the fact that the total area is distributed among more planes. Nonetheless, the number of successful runs using this approach is still smaller than those achieved by the CS-EA approach, which is able to solve every run with 2-shipwrecked people and all of the runs with 3-shipwrecked people, except for two, within the stipulated time window. In terms of time, and even though with 5 planes, as indicated above, the swarm based approach is marginally faster when it is successful, an increase in the number of planes leads to an improvement in the relative performance of the CS-EA which becomes as fast and sometimes faster than the swarm based approach in all cases (see Figure 12.c) where the average time differences are non-existent. These results show that, although the swarm approach may sometimes be faster in finding one individual than the CS-EA based approach, the latter is much more reliable in all cases. In addition, as the number of planes increases the time needed to find all the shipwrecked people is similar for both approaches when successful.

The same conclusions hold when analyzing the results of the cases where the electromagnetic signal was noisy (Figure 11.d, Figure 11.e and Figure 11.f). As in the noiseless scenarios, the swarm approach completes fewer cases than the CS-EA approach. Although the average time required to accomplish the tests is longer for both algorithms than in the noiseless scenarios, the reliability of

the CS-EA approach remains much better than in the case of the swarm based algorithm preserving the number of successful runs.

Summarizing, the performance of the CS-EA approach is more reliable and robust than the swarm based approach. This is because the search process of an evolutionary algorithm exploits the information of the search space provided by the planes more effectively. That is, the exploration vs. exploitation ratio is more balanced. This is not the case of the swarm based approach, which is an algorithm that is much more exploitative, making it more difficult to find several targets with a small number of planes. This highly exploitative search strategy leads to a fast convergence towards the first target or targets found. Thus, areas of the search space far from the starting point are hardly visited by the planes. On the other hand, the target population generated by the CS-EA guides the planes all over the search space, even to areas that had never been visited by the planes.

5 Conclusions

This paper proposes a new kind of evolutionary algorithms in order to address a set of problems that involve their application within the operational structure of real world multiunit systems. Thus, the applicability of Evolutionary Algorithms (EAs), and in particular that of a variant of the Differential Evolution algorithm, to real-time-real-world cooperation strategies for searching has been analyzed. This is a novel application of these algorithms that takes advantage of their distributed nature as well as of their exploration vs. exploitation structure, which is very easy to apply within efficient search procedures. Their efficiency for this purpose has been compared to that of swarm intelligence based approaches that, due to their simplicity and inherently distributed nature, have been widely used in this type of applications. This comparison has been carried out over a real example related to the use of UAV teams for finding several survivors of shipwrecks in a very large area. The results achieved show that the performance obtained by the swarm intelligence based approach decreases drastically with the increase in the number of targets. Even though this approach may be marginally faster than the CS-DE based approach in the simplest cases and just when searching for a single individual, the CS-DE based approach turns out to be much more reliable in all cases. In addition, as the number of planes increases, the time needed to find all the shipwrecked people is similar for both approaches when successful.

It should be remarked that, to the best knowledge of the authors, none of the works presented in the literature takes advantage of the search capabilities of evolutionary algorithms in real world search tasks. This is because this type of algorithms require that a fitness value be assigned to each individual (or point when searching in 3D space), which is often impossible or impractical to obtain in real applications, hindering the application of these metaheuristics directly to real world systems. Thus, we have proposed an adaptation of the evolutionary algorithms to be used in these problems where it is not possible to directly evaluate the points of the search space in the way proposed by common EA. The CS-EA approach developed here uses the information sensed from the environment to feed an evolutionary algorithm that generates a promising population which is utilized to guide the UAVs through the environment. The results provided by this approach are very promising as it outperforms the widely used swarm approach as the number of search objectives increases.

References

[1] T. Samad, J. S. Bay, and D. Godbole, “Network-centric systems for military operations in urban terrain: The role of UAVs,” in Proc. IEEE, vol. 95, no. 1, pp. 92–107., 2007.

[2] R. Sengupta and J. K. Hedrick, “Strategies of path planning for a UAV to track a ground vehicle,” in Proc. 2nd Annu. Symp. Auton. Intell.Netw. Syst., Menlo Park, CA, 2003.

[3] U. Zengin and A. Dogan, “Real-Time Target Tracking for Autonomous UAVs in Adversarial Environments: A Gradient Search Algorithm,” in IEEE Trans. Robot., vol. 23, no. 2, pp. 294–307, Apr. 2007.

[4] A. Feshchenko, “Bird's-eye perspective. A review of indigenous UAV designs”, Defense Express. English Version, No. 1-2, page(s): 52-59, February 2007.

[5] Y. Qu, Q. Pan, and J. Yan, “Flight path planning of UAV based on heuristically search and genetic algorithms,” in Proc. 31st Annu. Conf. IEEE Ind. Electron. Soc., p. 5, Nov. 2005.

[6] C. Zheng, L. Li, F. Xu, F. Sun, and M. Ding, “Evolutionary route planner for unmanned air vehicles,” IEEE Trans. Robot., vol. 21, no. 4, pp. 609–620, Aug. 2005.

[7] Zhou Chuan-xiang, Yan Jian-guo, Hu Shan and Qu Yao-hong "Navigation/Flight control system design of a UAV based on DSP and CPLD", Manufacturing Automation, 2006.
[8] M. Quaritsch, R. Kuschnig, H. Hellwagner and B. Rinner, "Fast Aerial Image Acquisition and Mosaicking for Emergency Response Operations by Collaborative UAVs", Proceedings of the 8th International ISCRAM Conference, Lisbon, Portugal, May 2011.
[9] E. Besada-Portas, L. de la Torre, J. M. de la Cruz, and B. de Andres-Toro, "Evolutionary Trajectory Planner for Multiple UAVs in Realistic Scenarios" IEEE Trans. Robot., vol. 26, no. 4, pp. 619–634, Aug. 2010.
[10] S. Mittal and K. Deb, "Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms," in Proc. IEEE Congr. Evol. Comput., vol. 7, pp. 3195–3202, 2007.
[11] Z. Jin, T. Shima, and C. J. Schumacher, "Optimal scheduling for refueling multiple autonomous aerial vehicles," IEEE Trans. Robot. Autom., vol. 22, no. 4, pp. 682–693, Aug. 2006.
[12] D. Howden and T. Hendtlass, "Collective intelligence and bush fire spotting," in Proc. Genet. Evol. Comput. Conf., pp. 41–48, 2008.
[13] Eberhart, R. C. and Kennedy, J. "A new optimizer using particle swarm theory". Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995.
[14] K. Han, J. Lee, and Y. Kim, "Unmanned aerial vehicle swarm control using potential functions and sliding mode control,". Proceedings of the I MECH E Part G Journal of Aerospace Engineering, vol. 222, No. 6, pp. 721-730, 2008.
[15] P. Dasgupta, "A Multiagent Swarming System for Distributed Automatic Target Recognition Using Unmanned Aerial Vehicles," IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 38, no. 3, pp. 549–563, May 2008.
[16] Y. Altshuler, V. Yanovsky, I. a. Wagner, and A. M. Bruckstein, "Efficient cooperative search of smart targets using UAV Swarms," Robotica, vol. 26, no. 04, Feb. 2008.
[17] Banks, A., Vincent, J. and Phalp, K. T.,"Particle Swarm Guidance System for Autonomous, Unmanned Aerial Vehicles in an Air Defence Role". Journal of the Royal Institute of Navigation, 61, pp. 9-29, 2008.
[18] G. Varela, P. Caamaño, F. Orjales, A. Deibe, F. López-Peña and R. J. Duro, "Swarm Intelligence based Approach for Real Time UAV Team Coordination in Search Operations", Third World Congress on Nature and Biologically Inspired Computing, pp. 372-377, Research Publishing Services, 2011.
[19] Lee, Dong-Seop, Gonzalez, Luis F., Srinivas, K., and Periaux, Jacques "Robust evolutionary algorithms for UAV/UCAV aerodynamic and RCS design optimisation". Computers & Fluids, 37(5), pp. 547-564, 2008.
[20] Nikolos, Ioannis K., Valavanis, Kimon P., Tsourveloudis, Nikos C. and Kostaras, Anargyros N., "Evolutionary algorithm based offline/online path planner for uav navigation", IEEE Transactions on Systems, Man and Cybernetics, pp. 898-912, 2003.
[21] Hansen, Nikolaus, and Andreas Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation." In Evolutionary Computation, Proceedings of IEEE International Conference on, pp. 312-317. IEEE, 1996.
[22] Storn, Rainer, and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." Journal of global optimization 11, no. 4, 341-359, 1997.
[23] Gulshan Singh and Kalyanmoy Deb, Dr. "Comparison of multi-modal optimization algorithms based on evolutionary algorithms." In Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06). ACM, New York, NY, USA, 1305-1312, 2006.
[24] Epitropakis, M. G., Plagianakos, V. P., and Vrahatis, M. N. "Finding multiple global optima exploiting differential evolution's niching capability". In Differential Evolution (SDE), IEEE Symposium on (pp. 1-8). IEEE, 2011.

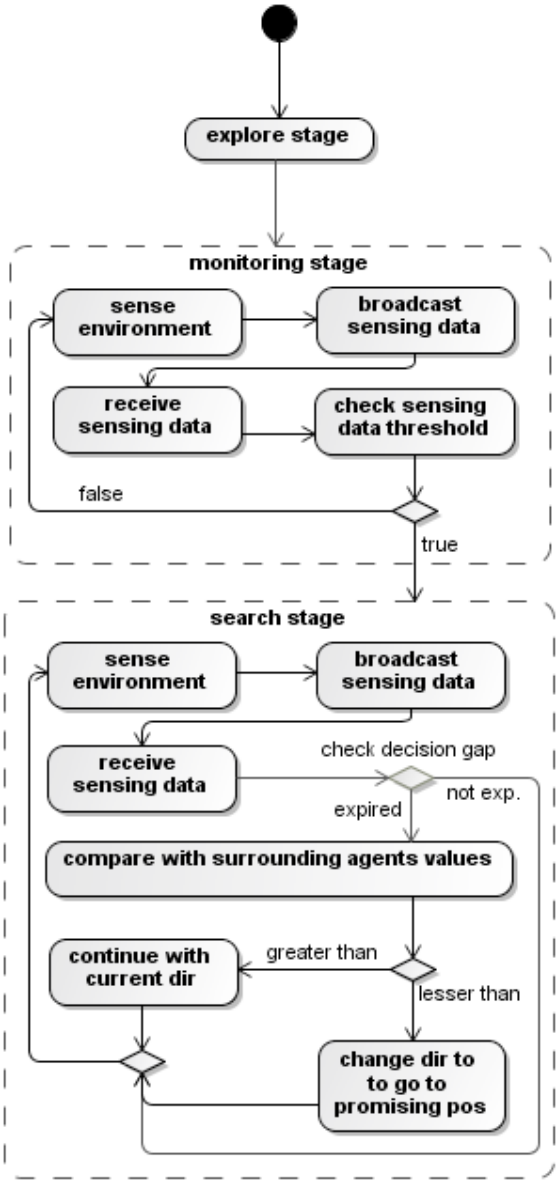

Figure 1. Agent behavior using the swarm strategy flow diagram.

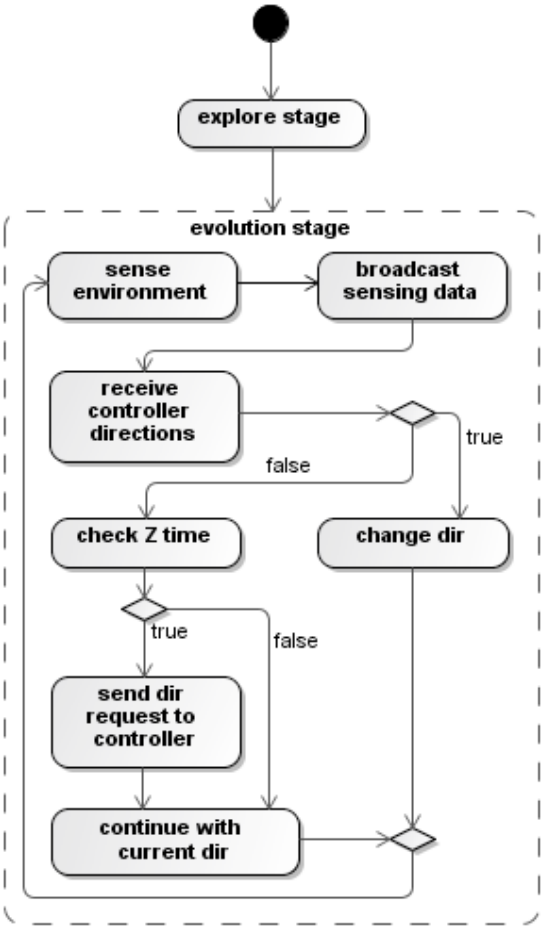

Figure 2. Agent behavior using an evolutionary approach flow diagram.

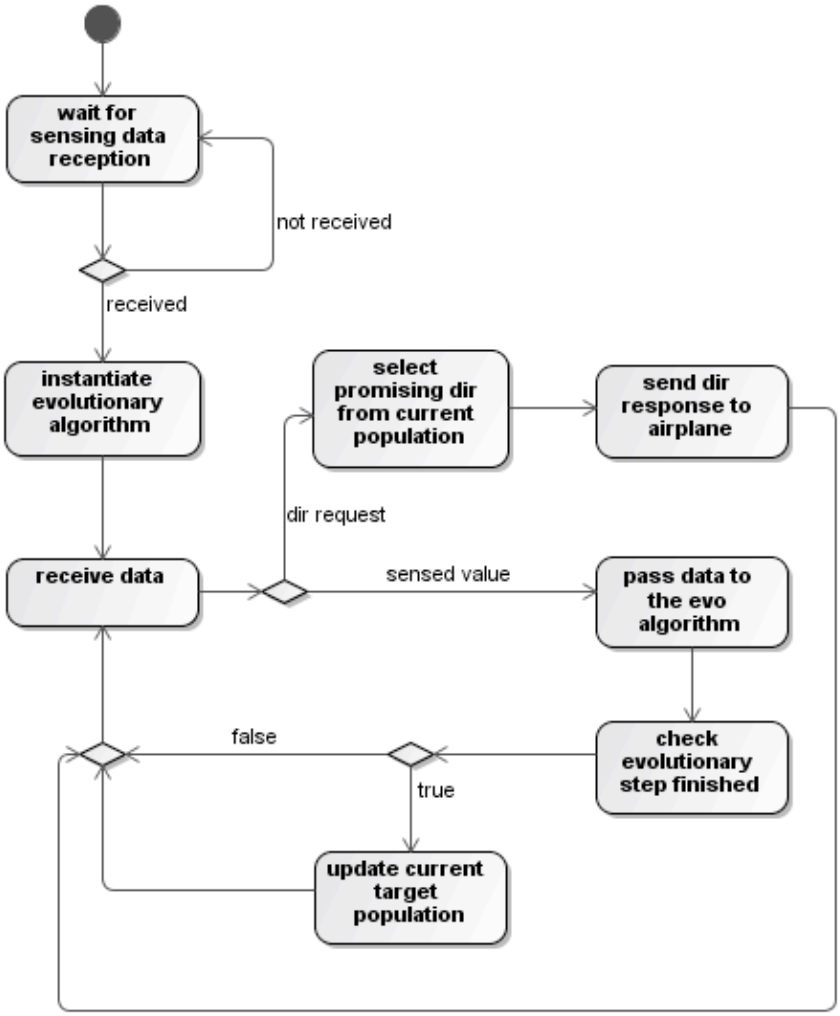

Figure 3. Evolutionary approach controller flow diagram.

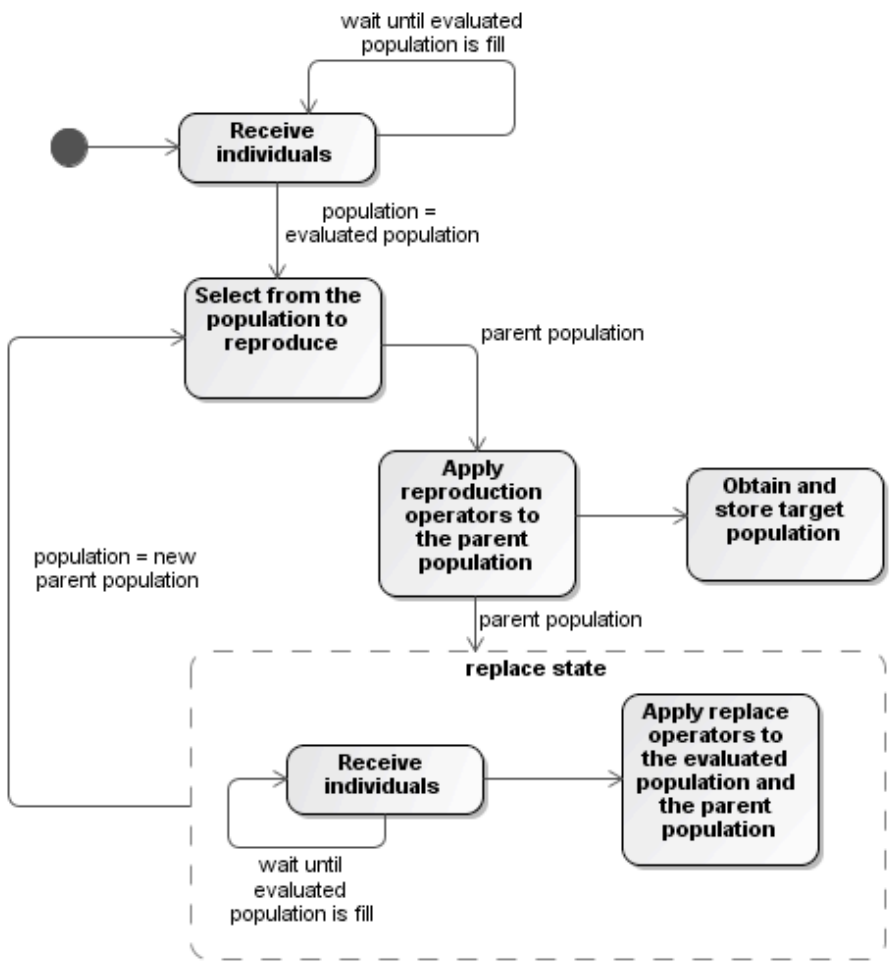

Figure 4. CS-EA algorithm flow diagram.

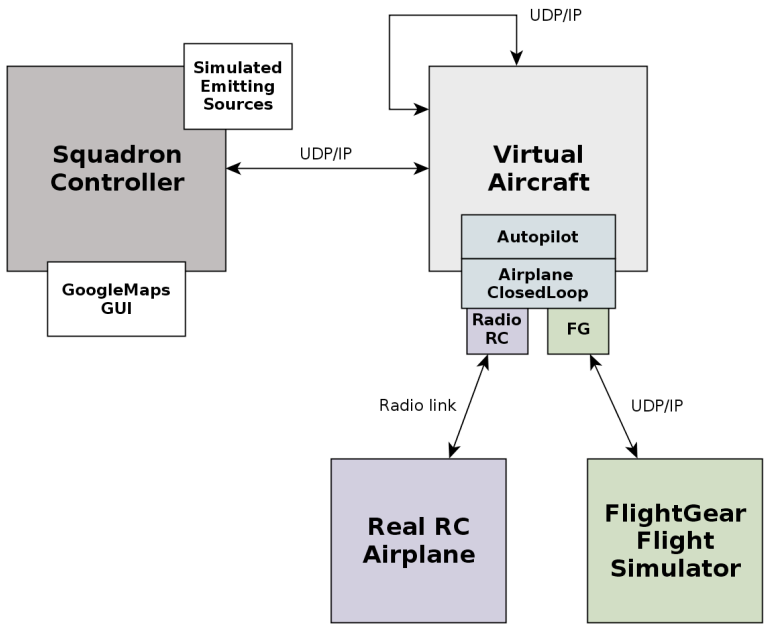

Figure 5. UAV team simulator block diagram.

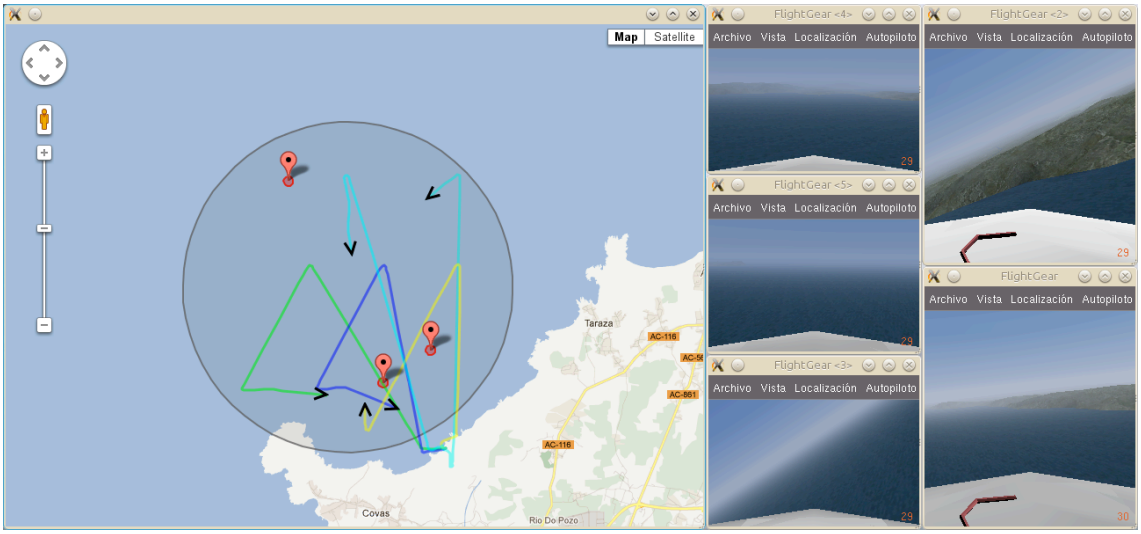

Figure 6. Snapshot of the 3D simulator with five planes.

Figure 7. Some of the UAVs that make up the team.

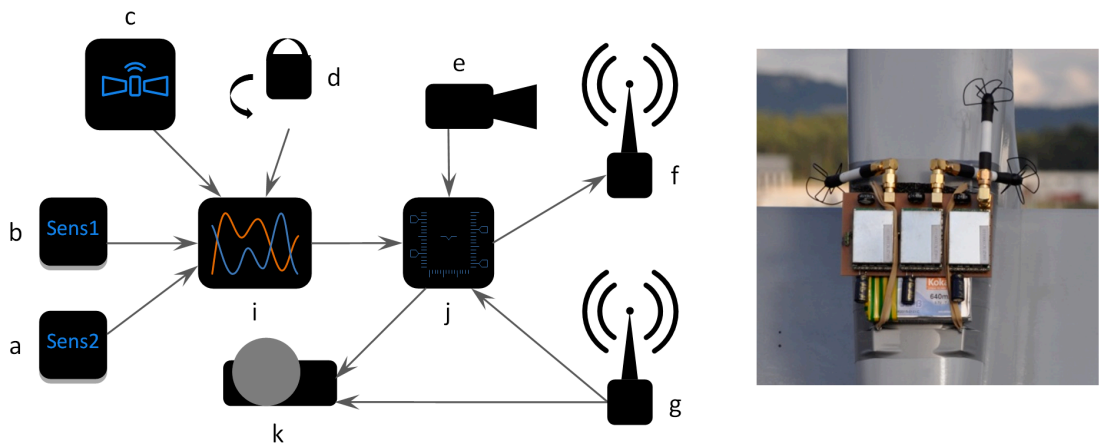

Figure 8. Left: Schematic diagram of the electronics within each airplane. Right: sensor attached to the plane.

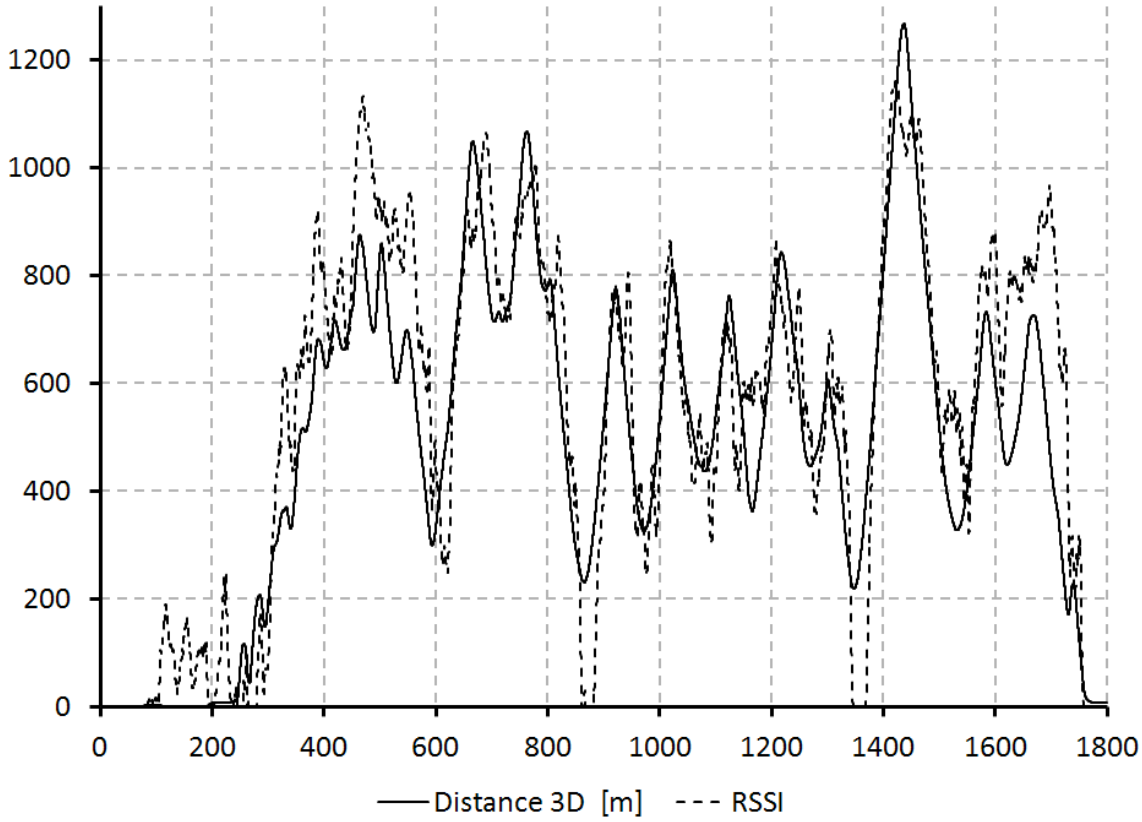

Figure 9. Correlation along time of flight between RSSI and 3D distance for one of the test flights.

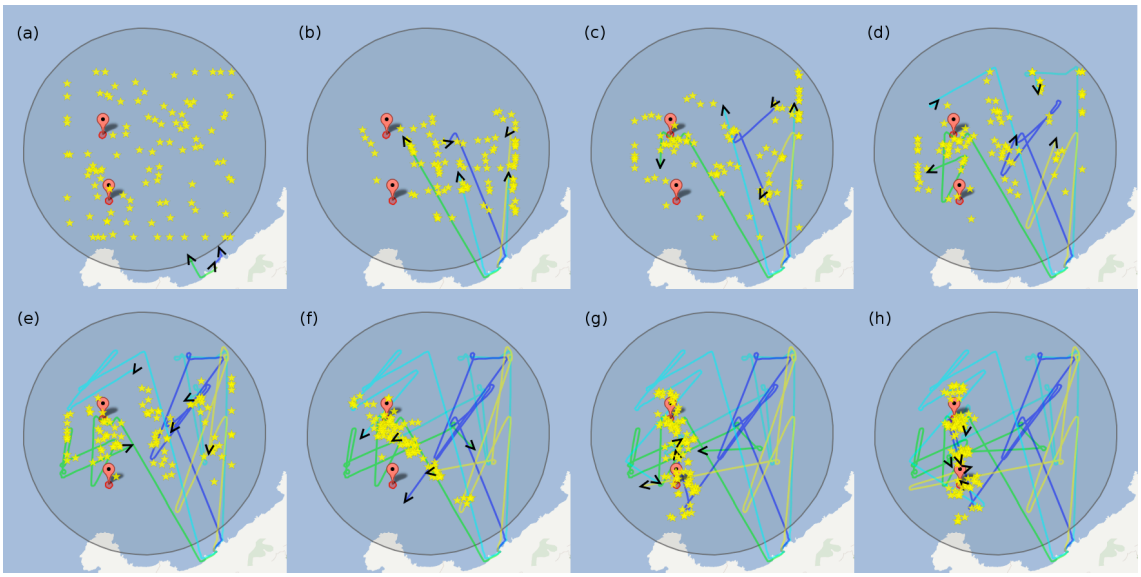

Figure 10. Sequence of snapshots from the simulator while running a simulation using the CS-EA algorithm, 2 shipwrecked people (circles) and 5 airplanes (arrows). The stars represent the target population provided by the algorithm in a given generation and the lines the trajectories followed by the airplanes.

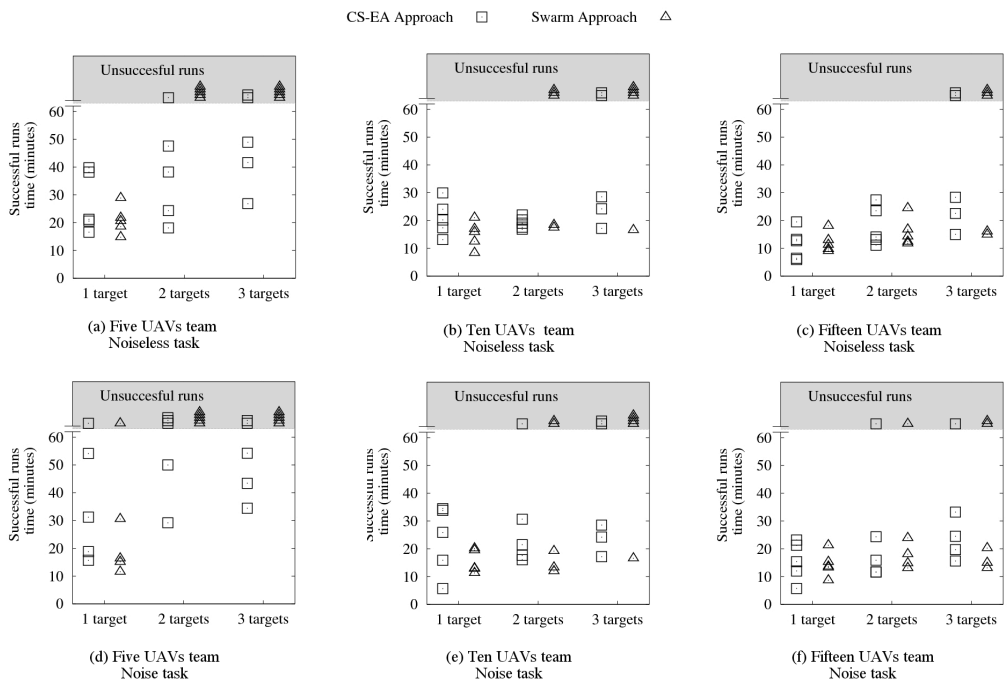

Figure 11. Results provided by the two approaches in the different test scenarios. From left to right, each graph shows the results for different team sizes: 5, 10 and 15 units respectively. Runs with no noise in the electromagnetic signal are shown in the top row while those where the signal is noisy are shown in the bottom row. Triangles correspond to the Swarm based algorithm and squares to the CS-EA.

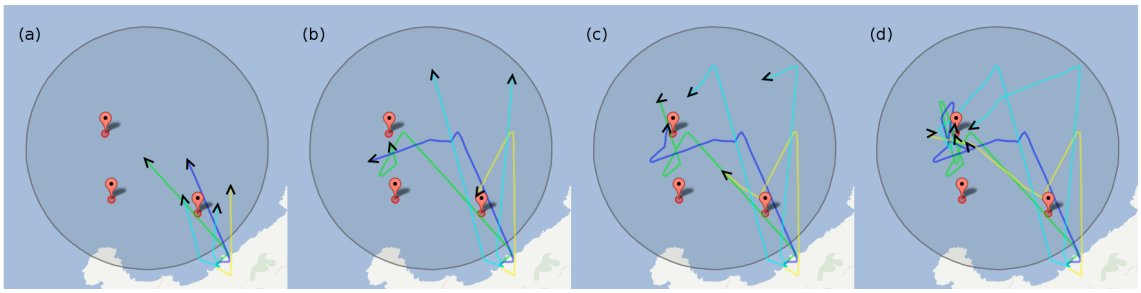

Figure 12. Sequence of snapshot from the simulator while running a simulation with the Swarm based algorithm. The case considers 2 shipwrecked people and 5 airplanes.